Assignment 3

Description

In this Assignment, you need to implement four classes: `Student`, `Course`, `Grade`, and `GradeSystem`.

- The class `Student` represents a student and contains attributes such as student name and student ID.
- The class `Course` represents a course and contains attributes such as course name and course ID.
- The class `Grade` represents a grade, and a grade belongs to a student who chooses a course, so the grade object needs to include the courses and student objects.
- The class `GradeSystem` represents a grade system. There are student list, course list, and grade list in the grade system. Through these lists, users can do various operations such as querying student information, querying course information, and querying grade information.

Problems

- Problem 0: Greeting (for demo purpose) [Easy, 0 marks]
- Problem 1: Design a class named Student [Easy, 15 marks]
- Problem 2: Design a class named Course [Easy, 15 marks]
- Problem 3: Design a class named Grade [Easy, 20 marks]
- Problem 4: Design a class named GradeSystem [Medium, 50 marks]

Notes for this assignment

1. You should submit all source code directly into your OJ system. Do not compress them into one archive.
2. No Chinese characters are allowed to appear in your code.
3. No package statement.
4. The output must strictly follow the description and the sample in this document. Do not print anything in all methods.
5. You may need to use `String.format()`.
6. Do not call `System.exit(code)` in your code.

Problem 1: Student

Design a class named `Student` in `Student.java`

In this problem you need to submit: `Student.java`

`Student` describes each student. In class `Student` you need to define:

Fields:

```
private int sid
// Student ID. Each student's student ID is unique and is automatically generated by class Student. Starting from 1, each new student's ID will increase by 1.
```

```
private String name
// Student name
```

```
private static int studentCnt
// Initialized to 1, increased by 1 everytime a Student object is created
```

Constructor:

```
public Student(String name)
// Constructor, automatically generate student ID and generate student object according to the name provided by the user
```

Methods:

```
public int getSid()
```

```
public String getName()
```

```
public void setName(String name)
// Modify student name.
```

```
public static int getStudentCnt()
// Get the value of StudentCnt
```

```
public String toString()
// When printing an object of this class, follow the format: "Student: %s, sid: %d" (without quotes)
```

Note:

- Do not define a `public void setSid(int sid)` method, as Student ID should be immutable.
- You can add other methods or attributes that you think are necessary.

Problem 2: Course

Design a class named `Course` in `Course.java`

In this problem you need to submit: `Course.java`

`Course` describes each course. In class `Course` you need to define:

Fields:

```
private int cid
// Course ID. Each course's course ID is unique and is automatically generated by class Course. Starting from 1, each new course's ID will increase by 1.
```

```
private String name
// Course name.
```

```
private static int courseCnt
// Initialized to 1, increased by 1 everytime a Course object is created
```

Constructor:

```
public Course(String name)
// Constructor, automatically generate course ID and generate Course object according to the name provided by the user
```

Methods:

```
public int getCid()
```

```
public String getName()
```

```
public void setName(String name)
// Modify course name.
```

```
public static int getCourseCnt()
// Get the value of courseCnt
```

```
public String toString()
// When printing an object of this class, follow the format: "Course: %s, cid: %d"
(without quotes)
```

Note:

- Do not define a `public void setCid(int cid)` method, as Course ID should be immutable.
- You can add other methods or attributes that you think are necessary.

Problem 3: Grade

Design a class named `Grade` in `Grade.java`

In this problem you need to submit: `Student.java`, `Course.java`, `Grade.java`

`Grade` describe each Grade. In class `Grade` you need to define:

Fields:

```
private Course course
// Course object corresponding to this grade.
```

```
private Student student
// Student object corresponding to this grade.
```

```
private float grade
// The grade (>=0).
```

```
private float gpa
// The GPA calculated by grade.
```

Constructor:

```
public Grade(Course course, Student student, float grade)
// Constructor, initialize student, course, grade and GPA.
```

Methods:

```
public static float calGpa(float grade)
// A static method to calculate GPA by grade.
```

```
public Course getCourse()
```

```
public float getGrade()
```

```
public Student getStudent()
```

```
public float getGpa()
```

```
public void setGrade(float grade)
// After set the grade, update the GPA at the meantime.
```

```
public String toString()
// When print an object of this class, follow the format: "sid: %d, cid: %d, grade:
%.1f, gpa: %.2f"
```

Notes:

- Do not define these setters:

  ```
  public void setCourse(Course course)
  ```

  ```
  public void setGpa(float gpa)
  ```

  ```
  public void setStudent(Student student)
  ```

- You can add other methods or attributes that you think are necessary.
- GPA conversion table:

GRADE	GPA
0~59	0
60~62	1.15
63~66	1.63
67~69	2.08
70~72	2.42
73~76	2.78
77~79	3.09
80~82	3.32
83~86	3.55
87~89	3.73
90~92	3.85
93~96	3.94
97~100	4.00

Problem 4: GradeSystem

Design a class named `GradeSystem` in `GradeSystem.java`

In this problem you need to submit: `Student.java`, `Course.java`, `Grade.java`, `GradeSystem.java`

`GradeSystem` describes each Grade System. In class `GradeSystem` you need to define:

Fields:

```
private ArrayList<Student> studentList
// The list of students
```

```
private ArrayList<Course> courseList
// The list of courses
```

```
private ArrayList<Grade> gradeList
// The list of grades
```

Constructor:

```
public GradeSystem()
// Constructor, innitialize studentList, courseList, gradeList. For a list with no elements in it, its size should be 0 and its reference should not be null.
```

Methods:

```
public ArrayList<Course> getCourseList()
```

```
public ArrayList<Grade> getGradeList()
```

```
public ArrayList<Student> getStudentList()
```

```
public boolean checkStudent(int sid)
// If a student is not in the student list, return false.
// Otherwise return true.
```

```
public boolean checkCourse(int cid)
// If a course is not in the course list, return false.
// Otherwise return true.
```

```
public boolean addStudent(Student student)
// If a student is in the student list (i.e. his/her sid is in the student list), return false.
// Otherwise add the student into the student list and return true.
```

```
public boolean addCourse(Course course)
// If a course is in the course list (i.e. its cid is in the course list), return false.
// Otherwise add the course into the course list, return true.
```

```
public boolean addGrade(Grade grade)
// If a grade satisfies:
//    - is in the grade list (i.e. a grade with the corresponding student's sid && the corresponding course's cid already exists in the grade list)
//    - its grade in the grade list is less than 60
//    - the new grade to be add is not less than 60
// Update the grade in the grade list by the new grade and return true.
// Else if the grade is not int the grade list && its corresponding student's sid already exists in studentList && its corresponding course's cid already exists in courseList, add the grade into the grade list, return true.
// Else return false.
```

```
public float gpa(int sid)
// Return the GPA of student with sid. The GPA is the average GPA of all the courses the student enrolled in.
// If the student has not taken any courses, return 0.
```

```
public float average(int cid)
// Return the average score of course with cid.
// If the course has no students, return 0.
```

```
public ArrayList<Grade> listStuGrade(int sid)
// Return an ArrayList<Grade> of the student with sid, the order of the grades is in accordance with the course order in the CourseList. Skip courses the student didn't take.
```

```
public ArrayList<Grade> listCouGrade(int cid)
// Return an ArrayList<Grade> of the course with cid, the order of the grades is in accordance with the student order in the StudentList. Skip students who didn't take the course.
```

Notes:

- You can add other methods or attributes that you think are necessary.
- Do not sort the lists.